D0 Note 3082

L1 Axial CFT Trigger Hardware and Firmware Design for the Baseline Trigger Algorithm

by Fred Borcherding

1. Introduction

A hardware implementation of the L1 Axial base line design has been developed and consists of four stages. First the signals from the VLPC's are discriminated and distributed to the trigger logic. Then the individual fiber signals formed into bins in each of the eight layers and compared to a list of roads to find track candidates. The third stage takes the track candidates which were found in phi and Pt bins and loads the found tracks into four output buffers. Each word in this buffer contains the phi and Pt bin address for one track. In the forth and final stage the four buffers are condensed into one and the information is sent to the other elements.

The axial trigger uses field programmable gate arrays, FPGA's, to implement the trigger logic. An advantage of using these devices is that the trigger logic can be programmed into them when they are in place in the detector. In principle they can be reprogrammed as required by changes in Physics interest or detector geometry at any time during a running period. FPGA's are also extensively used in the commercial market and market forces are expected to both drive down their price and drive up their performance.

The basic geometry of the tracker is discussed in several places.[1] Some of the basic features which are important for discussions of forming the trigger are presented here.[2, 3, 4] Each layer is made up of two single layers. The outer of these two layers is staggered by ½ a fiber so that there are no gaps, all tracks either pass through the inner, the outer or both layers. There are 8 doublet layers. The fibers are routed to the electronics so that the signals from all 8 layers for exactly 1/80 the of the detector go to one board. This 1/80th phi slice of the detector is called a sector. Each sector has 4 cells. A cell is the smallest phi slice which has a non-repeating geometry. A sector is 16 fibers wide at the innermost or A layer, increases by 4 fibers for each layer until it is 44 wide at the

8th and outermost layer, the H layer. A cell is 4 wide at the inner, 1 wider on each layer and 11 wide at the outer layer.

2. Finding Track Candidates

2.1 Doublet Finding

The first part of finding a track is to form a hit in each doublet layer. The individual fiber hits of each pair of singlet layers must be formed into a hit bin in the doublet layer. For the base line design the doublet bin size is the same as the fiber size of each layer. The doublet bin can be formed in the most basic manner possible, just an OR of an inner singlet fiber with one other outer singlet fiber. In equation form this is:

hl [k] = hi [k] OR ho [j];

where k and j are the k'th and j'th fibers on the ribbon. The indexes are such that the inner and outer fibers are adjacent, but whether the outer fiber is to the right or the left of the inner is arbitrary. This manner of forming a doublet doesn't distinguish if a particle transited one or both fibers. Also a particle which transits an inner fiber of one doublet pair and the outer fiber of the adjacent doublet pair will generate two doublet bins hit.

The doublet formation was expanded to eliminate the possibility of a single track forming two adjacent doublet bin hits. The doublet equation is then:

hl[k] = (NOT(ho[j -1]) AND hi[k]) OR ho[j];

Now if a track passes through hi[k] and ho[j - 1], hl[k] will be FALSE and only hl[k - 1] will be TRUE. Due to the architecture of the FPGA this modification does not take any more logic cells, LC's, but it does require more interconnects.

2.2 Track Finding

The eight doublet layer bins are then combined to form a track. The list of roads, which were found both analytically and with a special Monte Carlo, were translated into equations and loaded into the trigger logic. Roads were generated for a minimum Pt of 3 GeV and converted into about 1200 equations. The base line requires that all 8 of 8 possible doublet layers be hit for any equation to be satisfied. These equations are of the form:

T1013172227323945 = AL[10] AND BL[13] AND CL[17] AND DL[22]
AND EL[27] AND FL[32] AND GL[39] AND HL[45];

The several terms that share the same A layer doublet number, here 10, and the same H layer number, here 45, are then OR'ed together:

Trig_a10h45 = T1013172227323945 OR T10…45 OR …

These terms are then OR'ed together in groups that share the same H layer index but differing A layer indexes to form Pt bins. A strait line corresponding to an infinite momentum track drawn from the center of the detector and through the center of an H layer bin passes through just one A layer bin. That bin is defined as the zero offset bin for the H layer bin. The different Pt bins can then be defined with respect to the relative offset from the zero A layer bin. For the above example the zero A layer for an H layer bin value of 45 is 17. Therefore the offset for A layer bin 10 is -7. Table 1 gives a calculation of Pt for tracks of differing offsets from the H layer bin. Note that the width of a cell on the H layer is 11 and on the A layer is 4. As a result the bin center of the center H layer bin is exactly on the boundary between the 2nd and 3rd A layer bin. The zero offset bin for this case must be arbitrarily chosen.

Offset	Min	Mean	Max
0	18.00	21.40	
1	9.00	16.90	
2	6.50	11.00	21.00
3	4.50	6.80	10.50
4	4.00	5.00	7.00
5	3.25	3.90	5.00
6	2.75	3.30	4.00
7	2.50	2.80	3.50
8	2.20	2.40	2.80
9	1.80	2.20	2.50
10	1.80	1.90	2.20
11	1.60	1.76	2.00
12	1.50	1.62	1.80
13	1.40	1.53	1.60

Table 1. Track Pt as a function of bins offset. The first column is the bins offset, the second through forth columns are the minimum, mean and maximum Pt's for a track with the given offset. The numbers are from histograms of the Pt which were generated by a MC which generated tracks over all outer bins for a sector.

Four Pt bins were formed. The first bin was for offsets of 0 and 1, the second 2 and 3, the third 4 and 5, and the forth 6 and 7. The four 'negative' bins were formed with the same offsets.

The output from this stage is a matrix of pins which is 44 phi bins wide by 8 Pt bins long. Each pin on this matrix will be TRUE if a track was found or FALSE if it wasn't.

3. Serializing the Found Tracks

The track finding stage outputs a matrix of pins. This matrix is 44 long corresponding to the 44 phi bins by 8 wide corresponding to the 4 Pt bins of each polarity. The array must be searched in decreasing Pt order looking for any that are TRUE. As each TRUE pin is found the phi bin address and Pt bin address for that pin are loaded into a register. This is basically a serial problem which must be solved in parallel hardware. If it were done serially the process would take at least 44 x 8 = 352 steps. To get a result every crossing this processor would have to make 352 steps times the 27 MHz crossing frequency which requires a clock rate of over 9 GHz. Alternatively the problem can be solved using FPGA's using a tree structure with many

parallel branches in a very short time. However, this method requires significant resources.

The solution is formed in six steps which are done in the next set of FPGA's. In the first step groups of 4 phi pins are input into a set of 24 identical truth tables. The basic truth table is:

```
T0, T1, T2, T3 => c[3..0],a1[1..0],a2[1..0],a3[1..0],a4[1..0];

   0,  0,  0,  0  =>  H"0",  H"0", H"0", H"0", H"0";
   1,  0,  0,  0  =>  H"1",  H"0", H"0", H"0", H"0";
   0,  1,  0,  0  =>  H"1",  H"1", H"0", H"0", H"0";
   0,  0,  1,  0  =>  H"1",  H"2", H"0", H"0", H"0";
   0,  0,  0,  1  =>  H"1",  H"3", H"0", H"0", H"0";
   1,  1,  0,  0  =>  H"2",  H"0", H"1", H"0", H"0";
   1,  0,  1,  0  =>  H"2",  H"0", H"2", H"0", H"0";
   1,  0,  0,  1  =>  H"2",  H"0", H"3", H"0", H"0";
   0,  1,  1,  0  =>  H"2",  H"1", H"2", H"0", H"0";
   0,  1,  0,  1  =>  H"2",  H"1", H"3", H"0", H"0";
   0,  0,  1,  1  =>  H"2",  H"2", H"3", H"0", H"0";
   1,  1,  1,  0  =>  H"3",  H"0", H"1", H"2", H"0";
   1,  1,  0,  1  =>  H"3",  H"0", H"1", H"3", H"0";
   1,  0,  1,  1  =>  H"3",  H"0", H"2", H"3", H"0";
   0,  1,  1,  1  =>  H"3",  H"1", H"2", H"3", H"0";
   1,  1,  1,  1  =>  H"4",  H"0", H"1", H"2", H"3";
```

The input values are T0 through T3 and their possible values are the first four columns. The 16 possible combinations of the four input values are the 16 lines. The outputs are the 4 bit array c[ ] which is the count of pins that are TRUE (1) and four words each two bits wide that hold the phi address of the pin(s) that are true. At this point each group of four is in the same Pt bin so extra address bits for Pt bin are not needed.

At the end of this step there are 24 buffers each of which is 4 words deep holding from 0 to 4 phi addresses which are 2 bits wide. The remainder of the steps would be simply the combination of these 24 buffers by pairs except that 24 is not an even power of 2 and the address of the hits have to be extended. At this point they are only 2 bits wide and have to be extended to 6 bits to accommodate the 44 phi bins. The address also has to be extended by 4 more bits to accommodate the Pt bin address. Only 3 bits are needed for 8 Pt bins but 4 are used.

The next step combines these 24 four deep buffers by pairs into 12 buffers that are 6 deep. The code that does this is shown:

```
BEGIN
    CASE ca[3..0] IS
        WHEN 0 =>                   %  Register a is empty - move all %
            o1[ ] = b1[ ];
```

```
                o2[ ] = b2[ ];
                o3[ ] = b3[ ];
                o4[ ] = b4[ ];
                o5[ ] = H"0";
                o6[ ] = H"0";
        WHEN 1 =>                   %  Register a has one address %
                o1[ ] = a1[ ];
                o2[ ] = b1[ ];
                o3[ ] = b2[ ];
                o4[ ] = b3[ ];
                o5[ ] = b4[ ];
                o6[ ] = H"0";
        WHEN 2 =>
                o1[ ] = a1[ ];
                o2[ ] = a2[ ];
                o3[ ] = b1[ ];
                o4[ ] = b2[ ];
                o5[ ] = b3[ ];
                o6[ ] = b4[ ];
        WHEN 3 =>
                o1[ ] = a1[ ];
                o2[ ] = a2[ ];
                o3[ ] = a3[ ];
                o4[ ] = b1[ ];
                o5[ ] = b2[ ];
                o6[ ] = b3[ ];
        WHEN OTHERS =>
                o1[ ] = a1[ ];
                o2[ ] = a2[ ];
                o3[ ] = a3[ ];
                o4[ ] = a4[ ];
                o5[ ] = b1[ ];
                o6[ ] = b2[ ];
END CASE;

co[3..0] = ca[3..0] + cb[3..0];
```

This routine appends the contents of the second buffer to the end of the first while moving both into a third. It also makes the word count of the output buffer the sum of the two input buffers. The maximum buffer length is limited to 6, the information for any more found tracks is dropped from this step on. The track counter is 4 bits wide which gives it a maximum value of 16.

The routine which calls the above takes each of the 6 step 2 buffers and extends their width to 6 bits, adding the correct high address bits. The third step combines these 6 buffers which are now 6 bits wide into 3 buffers. These 3 are then combined into 2. At each step some buffers are appended onto the end of

others. They order of combination is controlled so that the contents are sorted in increasing phi bin order.

The final 2 buffers are one for the positive value of this Pt bin and one for the negative value. Both buffers are bit extended to hold the Pt bin address and the correct Pt bin address inserted. Then the negative buffer is appended to the positive to form a single buffer. (Which is in reality a positive track and which a negative is of course arbitrary.) The output of this stage is one of four sets of buffers holding the results for one Pt bin.

4. Distribution of Triggers

The forth and final stage is the sending of the found trigger information to other parts of the trigger. The data must be formatted into 6 words of tracks plus one header word and sent via fast serial link to the muon level 1 trigger. The data must be moved to a pipeline buffer for read out to the level 2 and level 3 on a level 1 trigger accept. The data must also be combined with the Central Preshower, CPS, information for formation of an electron trigger.

5. Hardware

The trigger hardware is being designed using FPGA's from Altera corporation. The Altera FPGA's were chosen for two reasons. First they continue to have the gate arrays with the largest number of available gates. Second they have a design and simulation package based on a PC running Windows which closely models the hardware, especially with respect to timing.

The finding of track candidates discussed in section 2 is done in a set of four FPGA's. Each FPGA is used to find the tracks in a single cell. Figure 1 shows the results of a simulation for the completed design. The data is input in four time slices and latched inside the FPGA. The output becomes stable for all channels 160 nsec later and remains stable for another 110 nsec. Anytime from 160 nsec to 270 nsec after input the output can be read. The first set of data input has all input fiber channels ON. This propagates through to all triggers being TRUE. For the second set of data only some of the input channels are ON and only some of the output triggers are TRUE.

The serializing of the data discussed in section 3 is done in another set of four FPGA's. Figure 2 shows the simulation for this design. New data arrives every 140 nsec. The output data is good after 90 nsec and remains good for about 50 nsec.

Figure 3 shows the data flow for the trigger. Data starts with the crossing at the top of the figure. The first stage is estimated to take about 80 nsec. Within this time after a crossing the VLPC converts the photons into electrons and the pick off chip discriminates the analog signal to produce a logic signal. These signals are then re-triggered and distributed to the next stage on the

home board and across the back plane, multiplexed in four time slices. The design for the signal re-triggering and distribution is still in progress.

The track finder logic for one cell requires just under 1800 LC's in the FPGA. This uses just over 60% of the Altera model 10k50 chip and about 35% of the larger 10k100. There are many things with would require the number of equations to be increased and few that would allow a decrease. The minimum Pt is 3 GeV, but there is physics interest in pairs of particles as low as 1.5 GeV. It takes about twice as many equations to lower the Pt threshold to 1.5 GeV as for 3 GeV. (It scales almost exactly as the inverse ratio of the Pt.) The equations used for this design assumed that the fibers were exactly placed on the detector. Our studies have shown that systematic placement errors starting as small at 50 um significantly increase the number of equations[5]. The beam spot for this design is assumed to be a point source. We know from accelerator that the beam spot will be about 50 um. The size of the beam spot has the same effect on the number of equations as fiber placement error[6]. Think of it as a fiber at r = 0. Also if D0 decides to build a displaced vertex trigger the subjective size of the beam spot must be made of the order of 1 mm or the trigger will veto non-prompt leptons. Also it is believed that while the efficiency of each doublet layer will be over 99.5% at the start of the run, this efficiency will drop with aging and radiation damage especially if we see high luminosity for much of the run. As the doublet efficiency drops below 99% the base line trigger which requires all eight layers drops below 90%. This can be remedied by only requiring 7 of 8 layers. A 7 of 8 trigger requires four times the number of equations. All of these argue that the track finder stage should be implemented in the largest FPGA's we can afford.

The serialization logic requires four FPGA's and uses just under 1900 LC's in each which is 66% of a 10k50. The requirements for this stage are not expected to change so we can have some confidence that the 10k50's are more than capable of performing this function.

Figure 1 - Simulation output for the track finding stage. Down the sheet are the various inputs, [I], to and outputs, [O], from the gate array. Across the sheet is the elapsed time in nsec. The data is input in four 20 nsec wide time slices and the output data is held for one cycle. Six complete cycles are shown. The data is always ready by 160 nsec and is stable until about 270 nsec from the start of data input.

Figure 2 - Simulation output for the serialization stage. Down the sheet are the various inputs, [I], to and outputs, [O], from the gate array. Across the sheet is the elapsed time in nsec. The data is input every 140 nsec and held for 120 nsec, similar to the expected output from the previous stage. The output is ready by 90 nsec and stable until 140 nsec after the start of input.

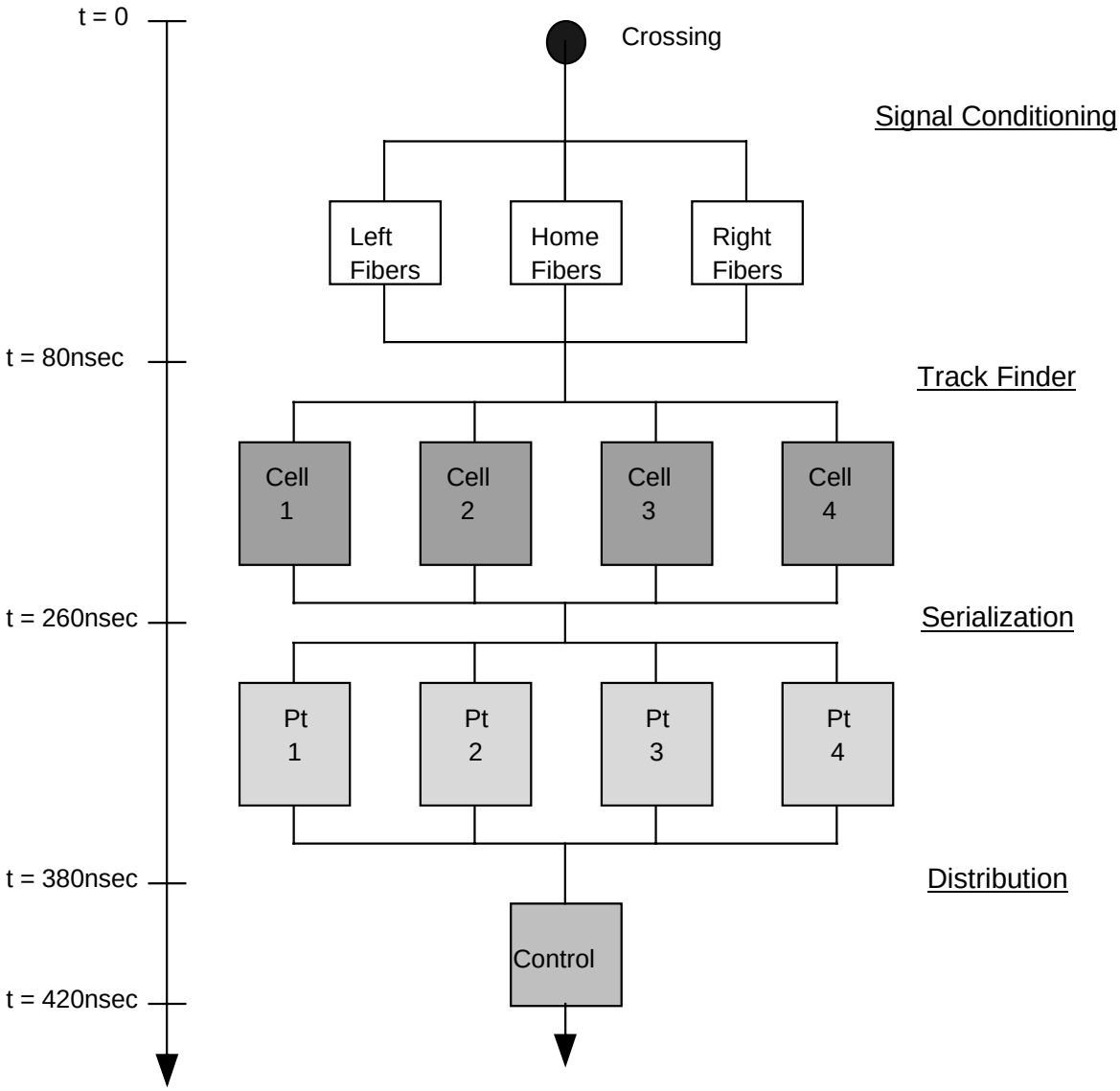

Figure 3 - Flow chart of for the level 1 trigger formation. The data flows from the top of the chart at t = 0, through each of the four stages of the trigger and is sent to the other detectors after about 420 nsec. Each of the filled boxes is an FPGA.The open boxes are the discriminator and other signal conditioning hardware.

[1] The 'Official' Detector parameter files are on the Web at: http://d0server1.fnal.gov/www/Upg_CFT/base_line.html

[2] D0 Note 2139, Electronics Design Specifications for the D0 Upgrade Scintillating Fiber Detector with a Level 1.0 Trigger, Alan Baumbaugh, Fred Borcherding, Marvin Johnson, Jesse Costa, Lourival Moreira, Sudhindra Mani, Steven Glenn and David Pellett, 19-July-1994

[3] D0 Node 2359, Level 1 Trigger Design for the D0 Upgrade Central Fiber Tracker, Fred Borcherding, 21-November-1994

[4] D0 Note 3058, D Zero Central Hardware Trigger Preliminary Implementation Studies of the "Base Line Design", R. Angstadt and Fred Borcherding, 15-August-1996.

[5] D0 Note 2504, A study of the Effects of Fiber Placement Errors on the Level 1 CFT Trigger, Fred Borcherding, 17-March-1995

[6] Addendum to: D0 Note 2504, A study of the Effects of Fiber Placement Errors on the Level 1 CFT Trigger, Fred Borcherding, 12-April-1995